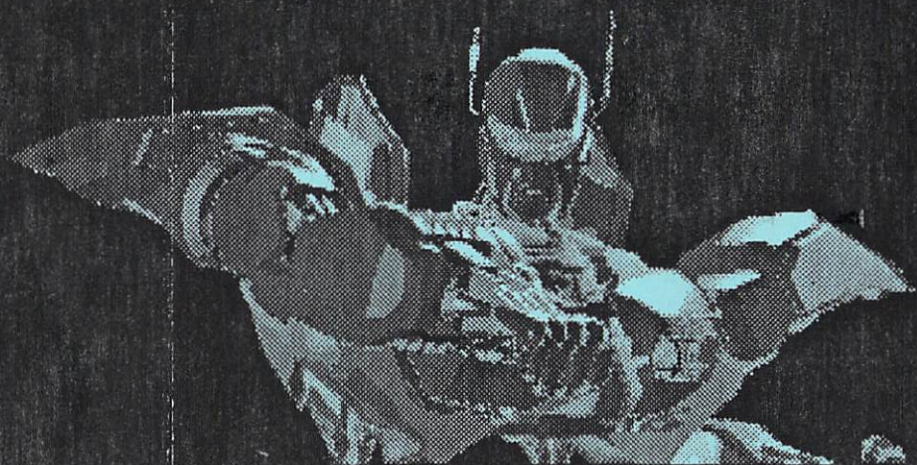


JUNE, 1994

Ohio Valley
AMIGA
Users Group

Get ready for the
Rise of the Robots



AMIGA CON

2 President's Palaver

3 About the Cover...

4 Editor's Message

5 Minutes of May Meeting

6 Telephone Numbers

7 From the Mind of
Jeff Johnson

8 AmigaDOS Synergy: An
Introduction to AmigaGuide

11 Tech Arena: Designing An
AmigaGuide Database

President's PALAVER

George Gibeau

Hmmm, not much to report this month. Nothing officially reported about Commodore yet. As you may have heard, our own Expert Services will be selling Amiga-DOS 3.1 to anyone who wants it, not just Picasso owners. The price for the A500/A2000 is \$125, and for the A3000/A4000 \$150. Maybe we can talk Scott into offering the club special pricing (hint, hint). So at least for now the Amiga is still alive.

We need volunteers for the upcoming ComputerFest, which will be held the last weekend of August. We need bodies, machines, demos, projects, etc.... If you would like to help out and require more information, please get in touch with me as soon as possible.

Nothing else really to report for this month, so get on with your reading of the rest of the newsletter.

[Yawn, yawn, heeoughhh - *Editor* ;-)]

About the Cover. . .

The *Rise of the Robots* is a game which is due out some time this fall. It should be available for CD³². The game features all 3D rendering using AutoDesk's 3D Studio software. Some animations serving as previews have been released in 3D Studio's FLC format. I used a Windows FLI/FLC player (AAPlay) to grab a frame to the clipboard. The image was then pasted from the clipboard into PhotoStyler, gamma corrected, lightened, given further contrast, and saved as a TIFF. Then I simply placed it into the PageStream document for your viewing pleasure!



Editor's MESSAGE

Dan R. Dennedy

Well, folks don't expect any more spectacular newsletters for a while. I recently upgraded to OS 3.1, and my Amiga puked. I've lost the ability to add the additional 32-bit fast RAM on my '040 accelerator into the system memory. I have 8MB on the board. 2MB AutoConfigures, but the additional 6MB requires a small program on startup to map in the additional memory which now chokes on 3.1. What's worse is that the board's manufacturer, Progressive Peripherals, is out of business—guess they weren't progressive enough. There is a simple `exec.library` function call, `AddMemList()`, which performs the desired operation. It all comes down to finding the base address of the memory. Now, how to figure that? I need to disassemble the "Init040" program, and figure out how the base addresses for the two memory banks are resolved. This can be pretty tricky. I don't know if I will be able to get anywhere, but I will try my damndest, I will certainly not go back to OS 2.1. Call me a masochist, but I accept the challenge and simply refuse to turn back!

You see, upgrading to 3.1 has allowed me to run AMosaic. At the very least I can use my Amiga as a NetStation, and do the rest of my work on the PC. Yeah, I hear you heckling at me, but those Windows applications are quite nice. For example, I received a demo version of a paint program called Fauve Matisse. It is a natural media type painting program (emulates things like oil paint, water colors, pastels, chalk, crayon, and quills on top of different canvas textures). Parts of images can be picked as brushes as any other paint program does, but they are maintained as free-

floating objects which can be later reshaped in any manner. In addition, one can apply any number of image processing functions to the brushes, as well as a separate alpha channel for each brush. Then, one can define the manner in which the brush merges onto the background image. Sounds pretty cool, huh? The demo version is a fully functioning greyscale version. The upgrade to the TrueColor version is only \$99. Now, if only TV Paint were like this....

This message goes out to the PC-SOBs: the conference call was amusing, cool, and all, but the bomb threats are definitely uncalled for. Don't be... well, I'll refrain from saying that in this forum.

Jeff Johnson has contributed quite a bit this month, and I appreciate his continued devotion to the newsletter. Thanks, Jeff.

Finally, we are concentrating on AmigaGuide this month. The feature articles were written by Treasurer Ed Hoffman as well as myself. Ed will provide an introduction to AmigaGuide at the Beginner's SIG. Time permitting, I plan on demonstrating the no-Net version of AMosaic during the regular meeting since it is a related topic and because it is just so awesome too. AMosaic uses the MUI (Magic User Interface) library; so, we could take a look at it at the same time and discuss its virtues and depravities.

Secretary's Minutes Of May Meeting

On May 31, 1994, at 20:00 hours, the good ship CSG OVAUG set sail with approximately thirty souls aboard. Approximate souls are something like virtual devices.

She left the dock scattered with the remains of the swap meet and raffle which preceded departure.

Officers' damage reports indicated a count, including Klingons, of 120 members total plus the addition of the video production company AMERICAM, \$265.00 in sustenance, and the BBS in perfect order.

Sighted by our VP as now residing in Cincinnati is Steve Worley [Is this correct? - *Editor*] of Apex Software. He offers OVAUG members a Club deal on their products Essence Vols 1+2, and The Forge. Please ask Drew Vogel for the details.

Put on your log the upcoming Computerfest the weekend before labor day. OVAUG and the other Amiga clubs did us proud last year. We have much continued vitality to prove, especially this year!!

Now for the fun part. . . the demos.

Drew Vogel played with Imagine 3.0 in front of everyone. He described it as a worthy tool for the creation of 3D objects and animations.

Subsequently, Drew demoed The Forge, a texture rendering engine. The two are available for a club special deal of approximately \$175.00. An approximate deal avoids a type mis-match with an approximate soul.

Dan Dyke, our Amiga missionary to Jordan ((Dan is presently in Jordan on another archaeological dig. He has taken his A1200, DCTV, and his camcorder for video documentation)) displayed slides made from data files created in TVPaint. The subject was a physical model of the Abila church photographed, scanned, composited in TVPaint, and converted back to film slides. Talk about multi-media! Dan's work, as always, was creative, handsome, and educational. He also showed other slides of his archaeological and computer collaborations.

With the demise of Commodore we need OVAUG and our flagship local Amiga store, Expert Services, more than ever. OVAUG and Expert are healthy; let's continue to support both!

(continued from page 14)

Step four is to construct the hyper links. Work on one document at a time comparing it against your final list of document nodes with keywords. If a keyword appears in the text of the current document, place the link within the text. Otherwise, make a list of links at the bottom of the document with a heading of "See Also:." This step is very labor intensive but is really the only step which actually makes a hypertext system.

Step five is to test and debug. Once that is done, the database is finished, congratulations!

CONCLUDING REMARKS

The construction and maintenance of a hypertext database really is a lot of work. However, if the database conforms to the above model, scripts and small programs can be written to aid the process. Using a good text editor, ARexx scripts can be written to do searching, to insert the AmigaGuide statements which provide the templates for each node and link, to create the list of document nodes with keywords, and to create a tree diagram. Furthermore, some machine learning algorithms, such as clustering, can be written which will automatically build the menu hierarchy based upon the keywords. Another linear search routine can be written to automatically build the hypertext links. A GUI interface would be nice for maintaining the database by allowing the designer to drag-n-drop documents to move them in the tree while automatically adjusting the menu and linear-browsing links. The GUI interface should also allow the drawing of lines to connect documents to each other.

Okay, so now everyone should be writing their own databases, and we should expect every new application released or upgrades to provide on-line help using AmigaGuide—yeah, right!



President, George Gibeau
721-1214
Vice-President, Drew Vogel
232-6439
Secretary, Jeff Johnson
651-1199
Treasurer, Ed Hoffman
521-3424
Newsletter Editor, Dan Denny
574-3125
Technical Advisor, Brick Eksten
371-9690
Club Librarian, Scott Bennett
371-2603
BBS Officer, George Gibeau
721-1214
BBS Phone Number (8N1)
241-8817

From the Mind of Jeff Johnson

BOOKLET FORM DOCUMENTATION FOR DTP

Making the booklet form for the newsletter [e.g. the format of this newsletter - *Editor*] was a "direct approach" proposition. I folded three, four, or five 8-1/2" x 11" sheets together, stapled them, and numbered the pages. Then they were taken apart and the page numbers on each side of each sheet was typed into ProPage document pages. HighTek, no???

I've recently been working to put my left wing philosophies of life, I call 'em the "Jems," into booklet form to publish. It is up to around 50 pages, and the direct approach, although it would be OK, is a one shot deal. The booklet interleave changes if more pages are added.

So... figuring out a logical approach appeared to be inevitable. The form, of course, is printed on front and back, stacked, stapled and folded. The pages are left and right on each half of the horizontal sheet. How to arrange these sides and pages into a DTP document conveniently printed?

Let's call each booklet page (half of a full page) a "book" page. Let's call each full document page a "doc" page. Each physical sheet of paper gets printed front and back. We'll call these "front" and "rear" doc pages. This newsletter is in this form.

Since desktop printers do one side at a time, the sheets must be turned over and fed through again. When doing the newsletter, I printed all copies of one side, and then all copies of the reverse side. This was OK with a few different pages of many each. It would be awkward for a few copies of many different pages.

So, let's arrange a scheme allowing all fronts in series, then all rears. This will be convenient for any number of copies of any number of pages. For example, if there are 50 doc pages and 6 copies are required, print pages 1-25 six times. Turn over the stack and print 26-50 six times. Then collate, staple, and fold.

Now, how to figure the book pages on each doc page?? They are interleaved in a somewhat confusing manner. What is the mathematical relationship? Start with the center doc page, the one you are looking at when laid out flat opened in the middle. Start from here in figuring book pages. Assume that the left half will be an odd numbered book page, and the right an even numbered book page. This will be true then of every doc page; odd on left and even on right.

Assume that the center page will contain the middle two book pages, say 49 and 50 of a 100 book page publication. Since there are two book pages for each doc page, there will be 50 doc pages, and 25 sheets of paper. Always use an even number of doc pages, since there will always be a front and rear, even if nothing is printed. We are counting from one rather than zero, so the last doc page of the first half is doc 25. On this page will be book pages 49 and 50.

When setting up a document in ProPage or any DTP software, create the required even number of doc pages (half of the book pages). Rotate them 90 degrees with two columns, not linked. Make a separate box on each page for the book page number (clone from one on the workboard). Number the book pages on each doc page according to the following formulae:

MaxDocPage# is the highest document page number. This is a constant determined previously when setting up the document. It will be half of the number of book pages. Use even values, as

there will always be two sides of the sheets of paper. Of course, the front and back covers will be counted as book pages under this scheme, but this simplifies computations.

To set up the page numbers on each DocPage:

Compute the: ODD (left) BookPage# = (DocPage# x 2) - 1

Compute the: EVEN (right) BookPage# = (DocPage# x 2) + (((MaxDocPage# / 2) - DocPage#) x 4)

Now consider how to find which DocPage to go to in order to find a specific BookPage to link the boxes for the articles.

To find an ODD (left) BookPage#: DocPage# = (BookPage# + 1) / 2

To find an EVEN (right) BookPage#: DocPage# = (((BookPage# x 2) - MaxDocPage#) / 2) - (((BookPage# x 2) - (MaxDocPage# x 2) / 4) x 3)

Substitute values for the variables and work from the inner parentheses out.

Link the boxes on the pages as required, and fill with text. Extra boxes can be added for sidebars, pictures, etc. An ARexx program to automate this could be written with the help of a six pack and a pizza.

Try it, it works. No more "hands on, hard way!!". This technique does simplify creating a booklet form document. Send the first half of the doc pages to the printer in the quantity you wish, then the second half on the rear.

C=NIL WHITHER AMIGA?

....so now that Commodore has died, what next? Rather than grumble and crab, of which there is a sufficiency, let's paint a scenario.

The author is an Industrial Designer by educational background. Industrial Design is a discipline and profession the core of which is the planning for manufacture of mass produced merchandise.

Assume OVAUG has purchased the Amiga

technology. How should we proceed?? Well, we must initially consider what went wrong.

As I have written before, the major problem has not been Commodore's advertising. The problem has been, and is critically so now, the exclusive ownership of rights to the technology, or SINGLE SOURCING. Apple's Mac is headed for the same iceberg as Commodore's Amiga, and for this same reason. The Mac's niche, desktop publishing, has been ported to the Windows environment so why buy a Mac rather than a clone except to upgrade? An MS-DOS/Windows/Intel personal computer has become a commodity appliance sold and supported by many competitors. Price and availability will be in the same category as soya beans and pig's ears. If somebody mis-manages his business, another will be more than happy for the sale. The customer need neither justify nor rationalize his purchase.

All of the little luxuries we are used to such as 30 character filenames, formatting a disk whenever we wish, GUI and CLI, even our loyal and family-like user base don't mean diddly dip. They never have except to us. I used to kid my friends that a burglar would have fun at my house. Imagine trying to fence my Amigas, Beta VCR's, and the keys to my Corvair!

Sure, Commodore should have delivered AGA two years earlier, advertised intelligently, etc., etc. The question is, to quote Miss Gooch, "What do I do now?"

Let's count our strengths:

- There is a worldwide group of millions of loyal Amigoids ravenous to forge on, or at least a good percentage of us. This is a hel-luva market with it's mouth wide open.
- Chips, boards, cabinets, and keyboards are chaff and dander. Merely continuing to manufacture A1200's and A4000's is no big deal.
- AmigaDOS is no embarrassment.

However, merely continuing as is will allow nothing more than a graceful exit.

(continued on page 10)

AMIGA DOS *Synergy*

An Introduction to AmigaGuide

by Ed Hoffman

A lot of documentation and other text files for the Amiga are now delivered in AmigaGuide format. What is AmigaGuide? It is the official, Commodore-developed hypertext system for the Amiga and is freely available to all Amiga users. Even though it has only been officially included with AmigaDOS 3.0 and later, Commodore has released an archive that contains AmigaGuide material for pre-3.0 machines. This is available on Fred Fish disk 920. Also, the OVAUG club BBS has the file AMIGAGUI.LZH in area 1 available for download, that includes the entire Commodore release for AmigaGuide with an installation program.

Hypertext is a context-sensitive text document, where keywords are highlighted in some manner (in AmigaGuide, they are made into on-screen buttons). These keywords are active; that is, when clicked on they direct you to another text section. In this way, a context-sensitive document can be easily built with links between the sections for the user to navigate. Instead of reading top down, or searching an index for your areas of interest, you can simply click on your areas of interest and get the information you want quickly. In addition, AmigaGuide has the capabilities to link to other AmigaGuide documents, show pictures, or play sounds.

An AmigaGuide document (typically identified with a filename ending in ".guide") is a standard text file with special embedded commands for document navigation. These

commands begin with "@", and allow for the identification of active keywords or phrases, links between the active words and other document sections, an index, and other data. In order to read an AmigaGuide file, you need a guide reader. For AmigaDOS 3.0+ users, the MultiView program has an AmigaGuide datatype, allowing the reading and automatic identification of AmigaGuide files. Other freely-available viewers include MinGuide, Most, and Man, in addition to the Commodore-supplied viewer, named appropriately enough AmigaGuide. To view the guide, the name of the guide reader you use should be used as the default tool for the document (by clicking once on the document icon and pulling down the Information menu pick from the WorkBench Icons pulldown. Alternatively, Cli/Shell users can type in "<guide reader> <guide name>" at any prompt, where <guide reader> is the name of the guide reader you use, and <guide name> is the name of the AmigaGuide file you are trying to read.

A sample AmigaGuide document looks like this:

```
@DATABASE "My.guide"  
@$VER Myguide 1.0 (6/19/94)  
@(C) "Copyright © 1994 EH Hoffmann"  
@AUTHOR "Ed Hoffmann"  
@WORDWRAP  
@INDEX MYINDEX
```

```
@NODE MAIN "My Guide"  
@NEXT FOREWARD  
My Sample AmigaGuide File
```

Table of Contents



```
@{" Introduction " link INTRO }
@{" Index " link MYINDEX }
@ENDNODE
```

```
@NODE INTRO "Introduction"
```

```
INTRODUCTION
```

This is a sample AmigaGuide file developed for the OVAUG Beginner SIG by Ed Hofmann. I borrowed this largely from the AmigaWorld article on page 43 of the February 1994 issue. There are only two nodes in this simple guide: the Main section, and the @{" Index " link MYINDEX }, besides this Introduction.

```
@ENDNODE
```

```
@NODE MYINDEX
```

```
This is the index.
@ENDNODE
•EOF•
```

The guide file begins with some identification information (all guide files must begin with the @DATABASE command), followed by the table of contents, which shows the active links available at the top level. The links (or nodes) are then listed. The navigation of the nodes is accomplished by the user in any order by clicking on the keywords (node names).

There are many utilities available to help construct AmigaGuide files from text files, extract text files from AmigaGuide files, automatically append an index to an AmigaGuide file, and other functions. Many of these are available from Fred Fish disks, AmiNet, and on the OVAUG BBS. I will be going over some of these at the Beginners SIG before the June 28 OVAUG meeting starting at 6:30 pm, at the normal monthly meeting location. I will also explain more of the basics of AmigaGuide, and go over the above sample guide file.

(continued from page 8)

Ultimately, there must be a compelling reason to buy, use, and expect support of a product. Amiga hardware and AmigaDOS has about the same future as MS-DOS/Windows and Mac/MacOS. They are all verging on senility just as Applet, AtariST, and C64 were a decade ago.

The answer?

1> Resume production of 680x0 based hardware. This will support developers, distributors, and consumers of the current technology.

2> Rather than continue development of AAA technology, hire the likes of Dave Haynie and the other sharpies to port AmigaDOS to RISC microprocessor technology and DSP (Digital Signal Processing) hardware. The goal should be the elimination of any proprietary hardware considerations. The proprietary nature of the Amiga has been it's true downfall, as previously discussed. With the speed of RISC and DSP, AmigaDOS itself and the functionality of the custom chip set can multitask and share hardware with whatever else the user chooses. Consider a screen or two of AmigaDOS, another of Mac, a Windoze on Mes-syDOS somewhere. No one need start over. AmigaDOS will fade from relevancy with it's head high, in stride with the others.

3> Before long all of us will open such a screen or window occasionally over a Hofmann Stout and giggle a little.



Designing an AmigaGuide Database

by Dan R. Dennedy

Learn to author your own AmigaGuide databases to keep track of your personal data and organize your notes. I've been prompted to get into this because I have numerous text files I've retrieved and made while perusing the various computer networks. I need to organize them. The AmigaGuide hypertext system can easily meet my needs as well as yours, surely. So be prepared to follow me. I am going to get into the technical details as well as talk about some abstract concepts. Read Ed Hoffman's introductory article first if you are new to the AmigaGuide.

A GRAPH THEORY PRIMER

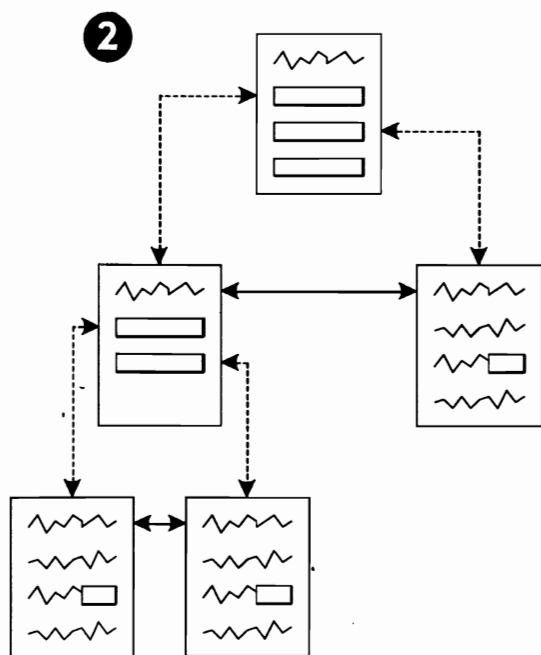
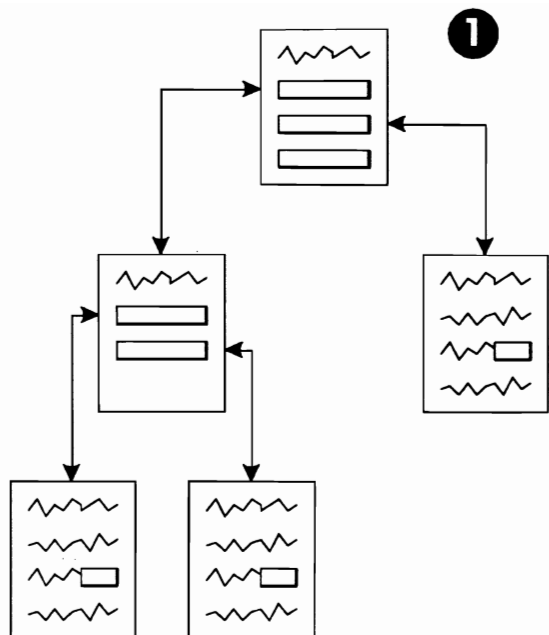
AmigaGuide is a hypertext format and browsing utility. Ed states that hypertext is a "context sensitive text document." While that is certainly true, allow me to provide another definition which will hopefully create a broader perspective for the concept of hyper technology. Typically data is organized in three different ways. First of all, it can be organized in a linear manner. A list is the most common representation of linearly organized data. In the case of text, one document simply follows the other, and possibly they are ordered by their titles. The next method of organization is by hierarchy. The tree is the most common representation of hierarchical data. We all are quite familiar with this concept. Some examples are family trees, business organization charts, filing systems (both paper and computer-based), books (sections and chapters), and menus. Hierarchy is all around us as we humans try to manage the

chaos. Unfortunately, hierarchy is often times too much order because navigating the hierarchy is simply inefficient. Think about it. The computer's menus are literally a drag sometimes because they are mouse intensive, hence the popularity of toolbars and toolboxes in modern applications. Furthermore, American corporations have recently realized the disadvantage of too much hierarchy. Consequently, they have decided to downsize and strip away the bureaucracy. This brings us to the last method of organization, the network. A network consists of a bunch of units, hypertext pages, which are related to a number of any other units. There is not the strict parent/child relationship of trees, and there is not the even stricter sequential order of linear data. A hyper-system embodies the network form of organization. A hyper-system can be hypertext, hypermedia, hyperdata, or any other suffix someone decides to add. The best way to visualize a hypertext system is to take a bunch of notes on index cards, throw them into a box, and mix them up. Then, pick out one card at random. Next, search through every card in the box to find out which ones are related to the current card. Now, do that for every card. By establishing links between the cards for every relationship, you have built a hypertext database! If you want to discover more about this topic, there is an area of mathematics devoted to it called graph theory which is a branch of discrete mathematics. Any computer science study relies heavily upon these areas (Trust me, I know. It's not the most interesting stuff).

The AmigaGuide system allows one to use all forms of organization simultaneously. The



"Browse <" and "Browse >" buttons let the user navigate in a linear fashion. Special pages can be constructed to serve as menus or documents. You are certainly familiar with those as most AmigaGuide databases tend to just revert into a menu-oriented (heirarchical) system. That's certainly disappointing, but it can be a very labor intensive to perform all of the cross-referencing. Nevertheless, it is definitely worth it, not to mention the fact that you can do some things to make the process easier.



THE HYPERTEXT MODEL

A model for constructing the AmigaGuide database is needed. Begin with a tree. The main menu is the single node at the top of the tree. The main menu branches down into submenus which branch into submenus or documents. The bottom of the tree (the leaves) consists solely of documents; otherwise, there is a menu which goes nowhere. (See Figure 1). Notice that there are several levels in the tree. By drawing horizontal lines across the nodes at each level, the linear organization emerges. (See Figure 2). To

make this real hypertext, all of the documents which are the leaves of the tree must be networked. (See Figure 3). This puts lines all over the place creating a mess, but that is advantageous. There is chaos in the order in our chaos.

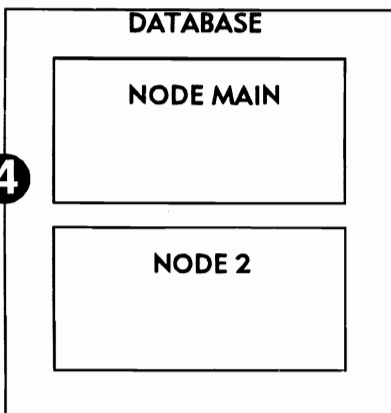
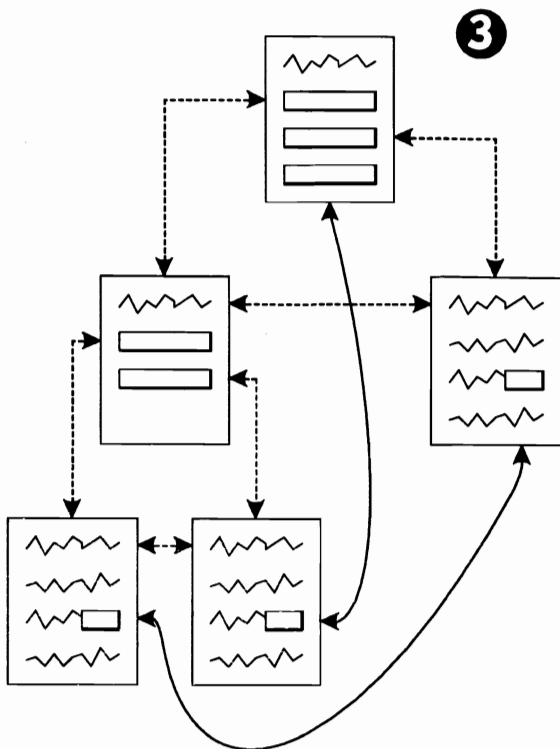
BASIC AMIGAGUIDE STATEMENTS

Before getting into the construction of actual databases, an understanding of the format of AmigaGuide databases is needed. There are only three constructs, or objects, in an AmigaGuide file: the database, the node, and the link. The file equals one database and begins with the line:

```
@DATABASE
"<database name>"
```

Note: angle brackets indicate arguments and are not to be included.

The database contains one or more nodes. (See Figure 4). A node may not contain other



nodes; so, they just follow each other in the file. Their order is not important except to the database designer. Nodes are denoted by:

```
@NODE <node name>
...
@ENDNODE <node name>
```

Simple enough. Note also that every node name must be unique.

The node contains textual information for the reader as well as zero or more links. Based upon our model, the nodes serve roles as both menus and documents. Therefore, the links transport the reader from menu to submenu, from menu to document, from document to document, or even from document to a menu. While all of the previous AmigaGuide statements must be placed

at the beginning of a line, the link statement can be placed anywhere within the node. The link statement is denoted by:

```
@{"<label>" <command> "<command arguments>"}
```

The <label> term actually forms the text on the button.

The <command> term identifies the type of link. The following chart breaks down the different links. No big deal; there's just a few of them:

Command, Arguments, Description

- LINK, <node name> <line number>, Go to the document identified by node <node name> and optionally scroll to line <line number>;
- RX, <file name>, Execute an ARexx script;
- RXS, <ARexx command>, Execute an ARexx command;
- SYSTEM, <AmigaDOS command>, Execute a program (just like entering a CLI command).

Just one more requirement; the first node in the database must be named "MAIN." That is all there is to an AmigaGuide database! All of the other AmigaGuide statements are merely supplementary. (See the AmigaGuide.guide file in Commodore's archive). At last, introducing the smallest AmigaGuide database ever:

```
@DATABASE "Smallest AmigaGuide"  
@NODE MAIN  
Hello World!  
@ENDNODE
```

I hope all is beginning to make sense. Take a break now if needed. Let it all sink in before suffering information saturation. We all have our thresholds; know yours!

THE HYPERTEXT MODEL IMPLEMENTATION IN AMIGAGUIDE

The best method for implementing a hypertext database according to the above model is to work bottom-up. Step one is to prepare all of the documents for inclusion. Place a line at the top of each document using the @TITLE statement. Then, enter all of the text. Next, identify keywords located in the document text, keywords which fully describe the document and may not be in the text, and keywords for any other related ideas you want. The keywords which describe the document are important in helping you build the menus so try to find words from the most specific to the most general. For example, a document describing a recipe for your pet (you decide!) might include keywords: biology, animal, pets, cats, tasty recipes, meat, feline, cooking, duties. Place the heading "KEYWORDS:" at the beginning of the list. Finally, choose a unique name for the node for this document and place the @NODE and @ENDNODE statements. This should be done for all of the documents to be included.

Step two is to construct the menus. To do this you need a list of all node names and the keywords for each node. The use of text searching utilities, such as grep, comes in real handy here (hint: search for the words "@NODE" and "KEYWORDS:"). The keyword list should be refined until a hierarchy is developed and until the keywords conform to it. Next, add nodes for all of the menus from the deepest levels up to the main menu (@NODE MAIN). Please be careful how many levels deep you go; five levels is probably the most. Again, work from the bottom up, so that you can enter the links immediately as you build the menus. Use the @TOC statement to link a node to its parent in the tree. According to our model, selecting the "Contents" button will go to the parent rather than all the way back to the main menu.

Step three is to construct the links which implement the linear browsing of nodes. Using the diagram for the tree, connect all nodes of the same level using the @PREV and @NEXT statements. That was easy.

(continued on page 6)

Ohio Valley Amiga Users Group
P.O. Box 428539
Cincinnati, Ohio
45242-8539



GOODWILL INDUSTRIES
OUR BUSINESS WORKS
FOR DISABLED PEOPLE

USA

